Technical Notes

FPGA TDC for MIPP Chamber System and Related Issues

6 January 2007

Wu, Jinyuan,
S. Hansen and B. Baldin

Wu, Jinyuan
Fermilab, MS 222
P.O. Box 500
Batavia, IL 60510 USA
E-Mail: *jywu168@fnal.gov*
Phone: *(630) 840-8911*
FAX: *(630) 840-2950*
Current version: 2007-01-06

Table of Contents

Index of Figures

Index of Tables

Document Revision History

Revision Number	Date	Description
1	Jan 6, 2007	Original version

1 Introduction

1.1 Basic Architecture

The basic architecture of the MIPP chamber TDC system is shown in Figure 1.1. The charged particle hit pluses are first amplified and compared with thresholds by discriminators to generate differential logic signals in the AMP cards. The hit signals are sent over short cables to the TDC cards mounted near detector. Each TDC card digitizes 32 channels of inputs.

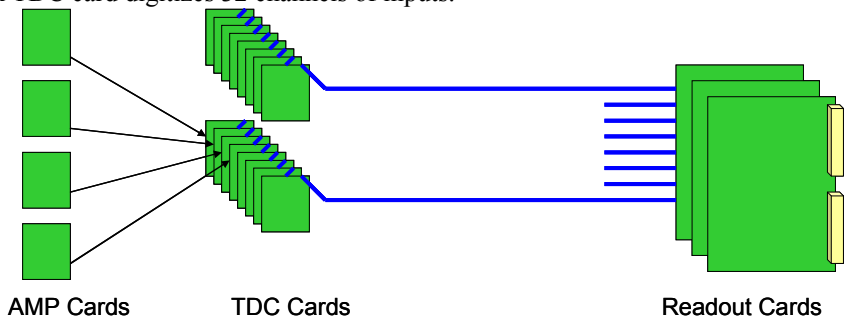

Figure 1.1
The MIPP Chamber TDC System Architecture

Each TDC cards has two RJ-45 connectors each containing 4 LVDS pairs, 2 inputs and 2 outputs. The TDC cards are chained together into a group of about 8 cards. Each TDC card group is connected to a 6U VMEbus based readout card via a CAT-5 cable. Each readout card provides 8 RJ-45 connectors for the TDC card groups. Under this configuration, each readout card severs 2048 TDC channels. For low data rate system, this is a reasonable density. The numbers are shown in Table 1-1.

	Each	Total
TDC card	32 channels	
TDC card group	About 8 cards	256 channels
Readout card	8 RJ-45 Connectors	2048 channels
Chamber System	About 8 readout cards	13K channels

Table 1-1
The MIPP Chamber TDC System

The primary advantage of this architecture is that the fragile analog and timing signals are kept to shortest possible transmission distances, which will not only increase reliability of the system, but also will save cost on the high quality cables carrying these signals.

There are special challenges for this architecture:

- Power supply to the cards.
- How to transmit clock, trigger and fast control information to the cards.
- How to provide generic computing abilities for thresholds, test pulse and other slow control and monitoring tasks.
- Error detection and notification.
- Data encoding and decoding.

These issues are to be addressed in this document.

1.2 A possible interconnection scheme

As mentioned earlier, each TDC card has two RJ45 connectors that allow the cards being connected into a chain structure forming a group. Each card group is served by a RJ45 connector on the readout card. One may further connect the end of two TDC card groups together as shown in Figure 1.2.

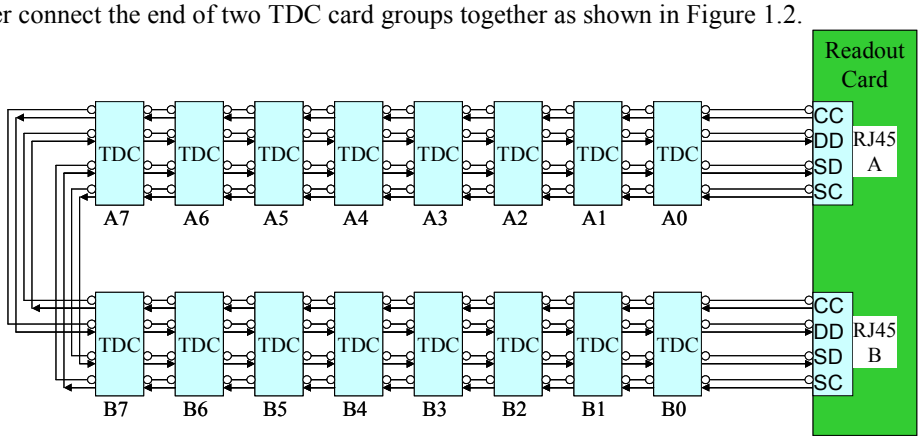

Figure 1.2
The MIPP Chamber TDC System Interconnection

With this interconnection scheme, it becomes possible to tolerate one bad interconnection during operation without having to replace cables when it is inconvenient. For example if the cable between card B5 and B6 is bad and the access to the detector is difficult, it is possible to reconfigure the cards so that B7 and B6 are served by RJ45 connector A along with cards A0 to A7. The RJ45 connector B now serves only cards B0 to B5.

The differential pairs in each cable are defined as the following.

- The signal CC is primarily a clock of about 26.5 MHz. Occasionally, fast commands like reset and trigger are carried in CC, which we will describe in the next section.
- The signal DD is the TDC data. It is encoded in 8B/10B standard at data rate of about 106 Mb/s.
- The signal SC is the slow control command. During normal operation, it is essentially quite. It is only used for FPGA configuring, parameter loading, error handling etc.
- The signal SD is the slow control command. As SC signal, it is primarily used for initialization processes while is essentially quite during normal operation.

If the data rate is very high, it is certainly possible to assign more differential pairs for data readout. For example, three pairs can be assigned for DD and one pair for CC and the traffic of SC and SD can be merged into CC and DD. However, for low data rate application, one pair of DD is sufficient. Remember that 106 Mb/s equivalent to a payload of about 10 MB/s while the VME backplane can only have a maximum data rate of 40 MB/s with D32 mode. Just one readout card with 8 RJ45 connectors will saturate the VME backplane if all DD payloads are to be readout.

The physical separation of SC and SD from the fast signals provides convenience for readout board design and initial debugging for the TDC cards.

1.3 The signaling of the clock and fast commands

The CC signal at most of time is just a plain clock of about 26.5 MHz, i.e., one period equals to two RF buckets in the Main Injector. A counter driven by the leading edge of CC in each TDC FPGA is kept as time stamp TS. An immediate question is that how to reset TS, or which clock cycle is the 0^{th} clock cycle. Also, if a clock cycle is missed or a event trigger is missed, how can they be detected. We discuss two options here, namely, the simplex scheme and the comprehensive scheme.

1.3.1 The simplex scheme

At the beginning of each run, the readout cards send 50% duty cycle pluses at 26.5 MHz to the CC differential pairs for several seconds. The phase-lock-loop (PLL) circuits and/or VXO oscillators in the TDC cards become stable. However, the time stamp counters are not synchronized yet.

At reset, a marker denoted as D0 is carried by the CC signal as shown in Figure 1.3. The D0 marker is a 25% duty cycle pulse followed by a 75% duty cycle pulse. Once D0 is detected by the decoder (after several cycles for pipeline delay), the time stamp counters TS and event counters EV in all TDC FPGA are reset which determines the 0^{th} clock cycle of the data taking run.

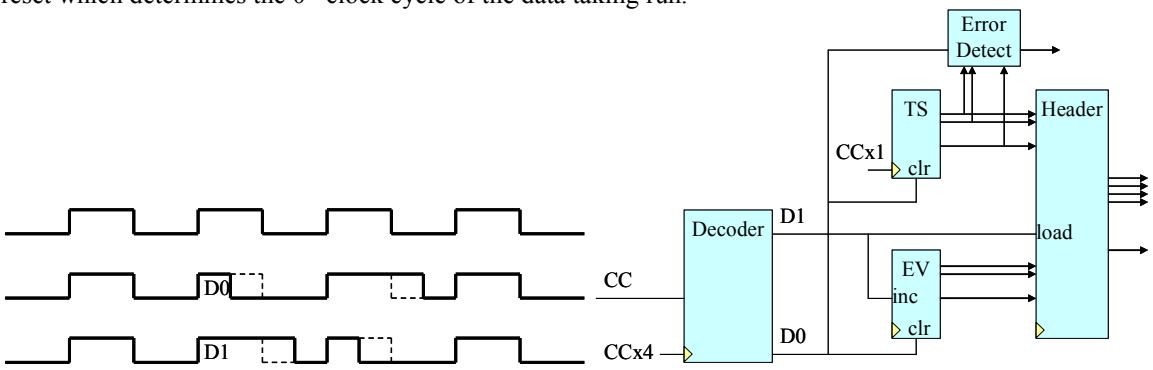

Figure 1.3
Signaling of the Clock and the Fast Commands

After reset, all TS counters should have same count after receiving the same number of leading edges of CC signal. The D0 markers are transmitted again when the lower bits of TS counters are to be rolled over. For example, the D0 markers can be sent every 294 cycles (i.e., 588 RF buckets, a Main Injector turn) with TS counter being mod 294 (or mod 7 and mod 42 if "super bunch" are to be counted). Each time when the D0 marker arrives, the lower bits of TS counter should be 0, if not, a miscount is detected and reported as an error. This error detection is provided but it should very rarely happen, practically never.

For trigger, a D1 marker, i.e., a wide-narrow sequence is carried in the CC train. When the D1 marker arrives, the values of the time stamp counter TS and the event counter EV are stored into a register, which then are merged into readout stream as part of the data header. The event counters EV in all TDC FPGA then increase by one preparing for next event. The time stamp at which the trigger processor issues a given event is known. If a mismatch between EV and TS is found in the header of the readout data, an error is detected and possibly can be corrected. Again this type of error, event counter miscounting should practically never happen but the error detecting ability is still provided.

1.3.2 The comprehensive scheme

If the users wish to provide more bits for more fast commands, the C5 scheme described in the next section can be employed. The CC pulse train is still a plain clock for most of time. When a command is to be carried, a DC balanced wide-narrow sequences are used, while all the leading edge are kept in the evenly spaced position. Unlike other coding schemes, in C5, the PLL or VXO driven directly by the leading edges of the CC train will not destabilize since the leading edges (not just transitions) are always at evenly spaced positions.

In C5 scheme, 5 pulses are organized as a DC balanced unit, carrying 24 combinations (or 4+ bits). Consider sending 3 groups or 15 pulses for each trigger command which takes 566 ns. One may assign 4 bits for command type, and 8 bits for redundancy check with the lower 8 bits of the event counters. We can further let the commands to be aligned with n*5 pulse boundaries. If the TDC FPGA finds the commands is not aligned with the boundaries, a TS miscounting error is issued. In this case, 5 combinations in the command type will be used to indicate clock cycle for the trigger, (or the starting point of the data taking window).

The bit rate efficiency of the C5 scheme is 25% of 8B/10B scheme. For example, a cable carrying 26.5 MHz CC signal should have a quality good for 106 Mb/s serial data. The data rate efficiency is trade for

simplicity and stability of the clock recovery in C5 scheme. In fact, the CC signal itself is a clock and no "recovery" is necessary, although the CC signal can be used to drive PLL and/or VXO.

1.4 The principle of minimum synchronization

It is a common "experience" in high-energy physics that a detector trigger/DAQ system needs more than one channel to distribute precise timing signals. For example, a system needs clock, fast reset, trigger and other fast control signals to perform various functions. In a lot of time, misalignment of these signals creates annoying sometimes confusing errors which could be avoided from the beginning.

In fact, entire system needs only to distribute one channel precise timing signal. All other commands such as triggers should refer to the master timing kept at the front-end cards, rather than being distributed via another precise timing network. We name this principle as "minimum synchronization".

The minimum synchronization principle does not exclude multiple precise timing distribution networks. In this case, the commands are still executed based on a single time standard. Redundant timing information obtained from extra timing channels are used for consistency checking and/or timing error reduction.

2 Some basic elements for the TDC cards

2.1 TDC in FPGA

Many time measurement functions in high-energy/nuclear physics experiments can be implemented in FPGA directly. There are two types of practical TDC structures one may choose from as shown in Figure 2.1.

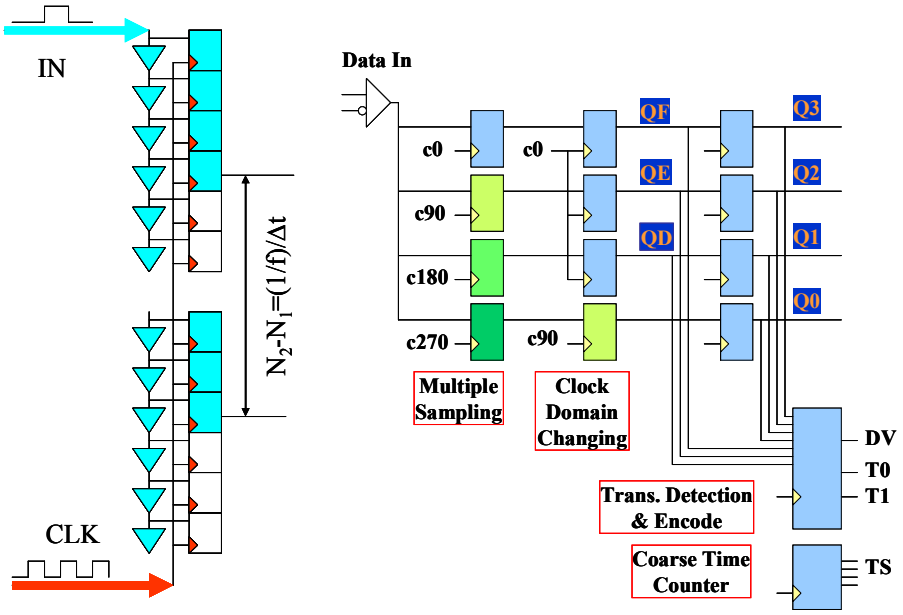

Figure 2.1
TDC structures in FPGA

The first structure shown in left uses chain structure like carry chain found in FPGA devices. The position of the input signal being registered represents the relative time difference between the input signal and the reference clock. The structure is commonly used in TDC ASIC chips except in ASIC chips the delay chain is adjusted by a control voltage that is derived by a feedback loop, so that the delay of each tap is a known constant. In FPGA, the delay of the delay chain is not controlled and it changes as the temperature and power supply voltage vary.

Instead of making compensation as in ASIC, the propagation delay of the delay chain is measured in real time. The delay line is designed to be longer than the clock period so that some input signals can be

registered twice by two clock cycles. From clock period and the number of taps between the two registered positions, the delay value of each tap can be estimated. Using this delay value, the actual arrival time of the signal can be found either offline or online inside the FPGA using a lookup table.

The second structure uses multi-phase clocks to register multiple samples of the input signal. If 4 phases of 250 MHz clocks are used, the input signal is sampled every 1 ns, which forms a TDC with 1 ns bin size (0.29 ns RMS). Note that the sampling interval is 1 ns but each register operates at 250 MHz, rather than 1 GHz. The clocks can be easily generated inside FPGA using internal phase look loop (PLL) blocks. The position of the input signal edge being sampled represents the arrival time and is encoded as lower two bits, T0 and T1 of the time value plus a data valid signal DV. The higher bits TS are generated with a coarse time counter. The coarse time, fine time and data valid signal is sent to later stages for further zero-suppression, buffering and packing operations.

The obvious advantage of the second structure is low resource usage and relatively low sensitivity on temperature.

2.2 Zero-Suppression and Time Stamp Assignment

Sending raw TDC or ADC data out of FPGA would normally require too big bandwidth. In fact, the ADC or TDC channel is not hit every clock cycle. Therefore, it is possible to suppress clock slot that contain no hit data.

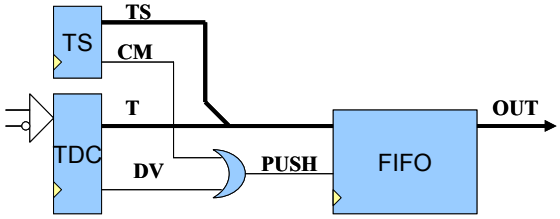

Figure 2.2
Zero-Suppression for Non-Trigger Front End

Consider a front-end digitizer without trigger as shown in Figure 2.2 in which all hits are sent to later stages. In the zero-suppression process, a time stamp (TS) must be attached to the hit data to identify which clock cycle the hit data is generated. In case of TDC, the TS bits are those of coarse time counter. The width of the TS is always a debate in nearly every experiment. A time stamp of k bits can represent up to 2^k clock cycles. If the TS is too short, the counter rolls over resulting in ambiguity of hit time in integer multiple of 2^k clock cycles. This is the similar problem as the Y2K bug. We will call the time period of 2^k clock cycles as a "centenary".

Increasing k is a possibility but it costs data link bandwidth. For example, a 32-bit time stamp can represent a time period of 85 seconds with 50 MHz clock. However, every hit must be attached with a 32-bit number while the hit data itself may be just a few bits.

Another possibility is to use shorter TS and send a "centenary mark" (CM) when the counter rolls over. For example, an 8-bit time stamp can be used with the TS counter counting from 0 to 254. The value 255 is reserved as the centenary mark. When the TS counter reaches 254, the FPGA insert a fake hit data in the data stream with time stamp value 255 if there is no real hit at this clock cycle. (If there is a real hit at TS = 254, the time stamp value is 254 to indicate that it is a real hit data.) The receiving devices use the centenary marks to increment the upper bits of the TS counter.

In this scheme, the total number of bits sent out the FPGA in T clock cycles can be written:

$$N = \frac{T}{2^k}k + Tfk$$

The parameter f is the hit rate, defined as number of hits per clock cycle. The results are plotted in Figure 2.3 for hit rates f = 0.005, 0.01 0.02 and 0.05 hits/clock cycle.

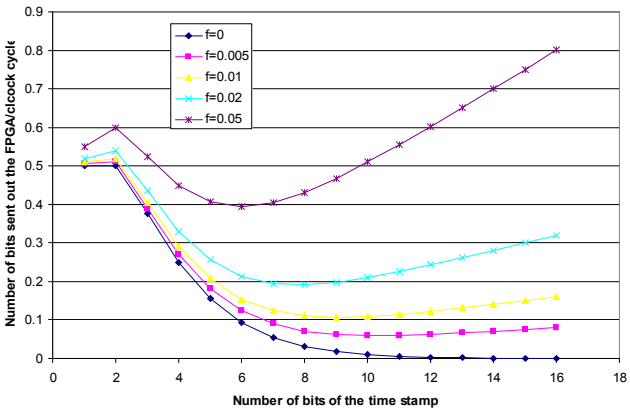

Figure 2.3
Data Output Rate from the FPGA

It can be seen that with hit rate of around 1%, the FPGA output rate is minimum when the number of bits used for time stamp is 8-12. Long time stamp is only reasonable when the hit rate is extremely low.

The choice of the time stamp also depends on other factors like time needed for the accelerator turn. For example, an accelerator turn in Fermilab Main Injector is 588 clock cycles at 53MHz. It is more convenient to choose mod 3, mod 4, mod 7 or mod 49 counters for corresponding bits of the TS counters.

2.3 Pipeline vs. FIFO

Pipeline and FIFO buffers are two popular types of memory organization methods utilized in high energy physics trigger and DAQ systems. The names may not reflect the actual properties of the two buffer types. In fact, the data stored in and retrieved out of a pipeline is also in the first-in-first-out fashion. In this document, we use pipeline to refer a buffer with constant store to retrieve steps, which can be visualized as a serial-in-serial-out shift register. A FIFO buffer is a storage which data can be pushed into and popped out with same data orders for push and pop operations. The pipeline and FIFO buffers are shown in Figure 2.4.

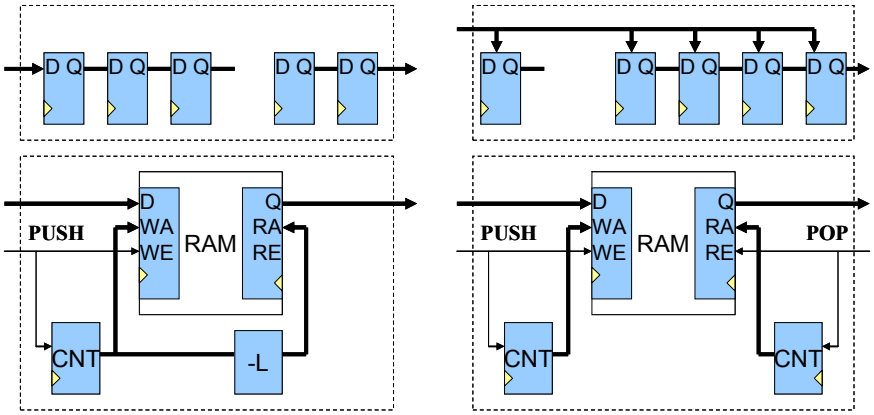

Figure 2.4
Pipeline and FIFO Buffers

In a FIFO buffer, the write address (WA) and read address (RA) are kept with two counters. The write enable (WE) signal is derived from the PUSH signal which also increases the WA counter. In the output side, the read enable (RE) signal is derived from the POP signal that also increases the RA counter. Some varieties of the FIFO may have logic to check for empty (i.e., WA=RA) or full (i.e., WA-RA= number of RAM words -1) conditions. Sometime, additional logics are added to prevent outside circuit from pushing into a full or popping out an empty FIFO.

The pipeline buffers can be viewed as shift registers but actually they are rarely implemented with shift registers chained up with flip-flops. The flip-flops are not efficient to store data not only in FPGA, but also in ASIC chips. Implementing long pipeline buffers with shift registers unnecessarily consumes large amount of silicon resource. Also, when data are clocked through flip-flops, transistors are turned on and off in all steps causing large power consumption.

The actual implementation of pipeline buffer is implemented similar as FIFO except the read address RA is derived from write address WA with WA-RA = L where L is the length of the pipeline. The RAM cell uses a lot less transistors than the flip-flop. After a data is stored in RAM, the transistors of the storage cell keep the on or off states unchanged until next time being overwritten, instead of changing every clock cycle as in flip-flops.

In high energy physics trigger and DAQ systems, pipelines are often used to store detector data for a fix number of clock cycles waiting for L1 trigger. The FIFO buffers, on the other hand, are used when the instantaneous data rate is considerably different from the average data rate such as in the zero-suppression process. A possible triggered front-end design that uses both pipeline and FIFO buffers is shown in Figure 2.5.

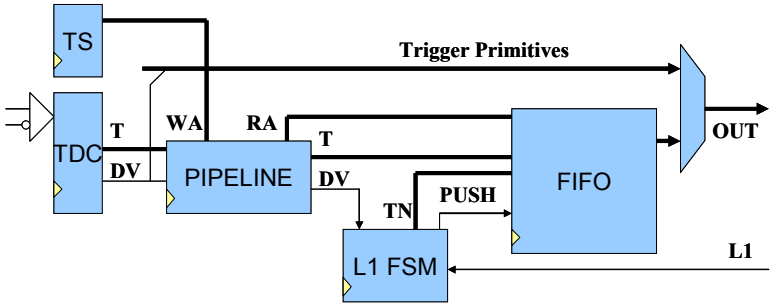

Figure 2.5
A Trigger Front-End

In this model front-end FPGA, the arrival times of the detector hits digitized in the TDC block with DV signal indicating that a valid hit is detected at the given clock cycle. The bits of T representing the arrival time of the detector hits along with the DV signal are written into the pipeline buffer every clock cycle, regardless these is a hit or not, and the time stamp TS is used as the write address WA. The immediate hit information, perhaps, just the DV signals of all channels are sent into the output data stream as trigger primitives.

Of course, the trigger primitives can also be more complicate. For example, the mean time of the arrival times of two adjacent channels can be used which represents the particle track hit time or event time (plus a constant delay) in drift chamber cases.

The trigger primitives are collected by the trigger/DAQ system and are used to generate a global level 1 trigger L1.

The L1 returns back to the front-end at the time when the hits from the trigger event are about to reach the end of the pipeline. Conducted by the L1 finite state-machine (FSM), non-empty data are pushed into the FIFO buffer. Now DV signal is used to derive the PUSH signal so that only non-empty data are pushed. However, additional bits must be included into the hit data. Usually, a trigger number TN or something similar is put into the trigger packet data header. For every hit, several bits of the read address RA that represent the coarse time must be added. The hit data pushed into the FIFO are sent out to the trigger/DAQ system when the output channel is not used to send the trigger primitives.

Also, it is not necessary to require the L1 return latency to be constant. The L1 trigger can be a multi-bit command and several bits in the command can be assigned as starting time stamp of the L1 window. The L1 FSM generates the corresponding read address RA that will ensure the hit data in correct timing window are collected. This scheme can be used in trigger systems when variable L1 trigger latency is necessary.

We shift our attention to comparison of the pipeline and the FIFO buffers. In the pipeline shown in previous example, many time slots contain no valid hits. It seems that zero-suppression should be done

right after the TDC, rather than after receiving the L1 trigger. In other words, it appears to be more economical to replace the pipeline in Figure 2.5 with FIFO.

However, several factors must be considered while choosing zero-suppression stage. First, zero-suppression process increases data word width since the time stamp must be added, while in pipeline buffer, the read address RA itself is the time stamp. Second, in the FIFO used for zero-suppression, a WA and a RA counter must be kept for every channel, while in pipeline buffer the WA and RA are common for all channels.

It is certainly possible to implement the fix latency pipeline with zero-suppression using FIFO. A block diagram is shown in Figure 2.6. When the hit data is pushed into the FIFO, the time stamp TS is also stored along with fine time T. At the output side, the TS value of the last hit is compared with the current TS. When the last hit is older than the predefined pipeline length, the POP operation is performed so that the RA in the FIFO points to the next newer hit.

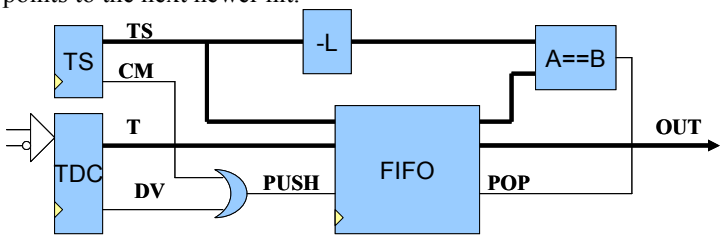

Figure 2.6
Implementation of Pipeline Using FIFO

It must be point out that the FIFO full is an error source in addition to other possible error sources in the entire front-end and trigger/DAQ system. In system with low hit rate, a sufficiently deep FIFO can reduce the probability of FIFO full error to nearly impossible. However, as long as the probability is not zero, an error code and all corresponding error handling processes must be implemented. To completely eliminate the possibility of the FIFO full error, the FIFO depth should be bigger than the pipeline length. If so, the FIFO uses more memory space than the plain pipeline without zero-suppression and therefore, there is no advantage to do zero-suppression at all in this situation.

2.4 Clock-Command Combined Carrier Coding (C5)

The FPGA TDC and commercial ADC allow the designers to place digitization functions in close proximity of the detector. When the digitization is done near the detector, delicate analog or timing signals will not need to be sent over long cables. However, necessary supports must be appropriately planned for the front-end digitization devices.

Obviously, the digitization FPGA must be clocked. At the beginning of a run, there may be some registers or parameters to be set in the FPGA that requires a means of command transmission. Before data taking, the first clock cycle needs to be marked so that the time stamp counter can be properly started. During normal operation, trigger acceptance commands that start data downloading are to be sent to the front-end for triggered experiments.

The "Clock-Command Combined Carrier Coding" (C5) scheme was developed to send commands especially synchronized ones like the first clock cycle marker with clock signal using a single link. In the C5 scheme, all leading edges of the pulses are separated with an equal distance as a regular clock signal while the data are encoded into the pulse width as shown in Figure 2.7. Please see [C5_IEEE.pdf] for details.

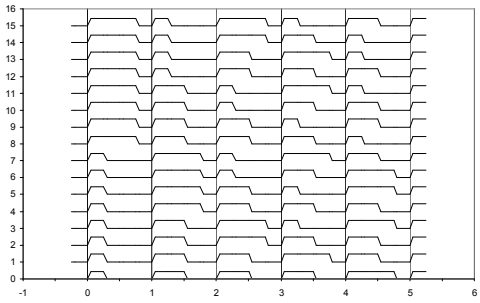

Figure 2.7
The C5 Pulse Trains

When cabling is limited, it is economical to combine commands which otherwise need separate links, into the clock channel.

It is a natural fear that carrying information in clock may cause jitters. In fact, the PLL block in the FPGA permits the different duty cycles used in the C5 scheme. Experiments show that there is no visible instability caused by carrying information in the input clock to the FPGA. As long as the potential jitter is lower than what required by application, it should not be an issue. For example, when a 1-ns bin size TDC is implemented in the FPGA, the RMS contributions to the measurement error for jitters up to 100 ps are practically negligible.

Also, commands for register setting and parameter loading are typically used at beginning of each run. The marker of first clock is only sent once before real data taking. The only possible commands being sent during data taking are triggers, which always happen after the event of interesting being stored in the pipeline.

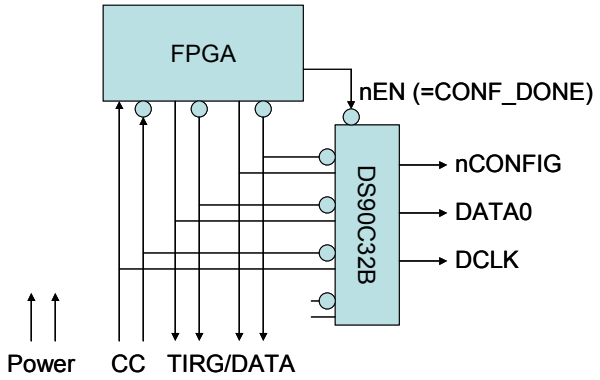

Figure 2.8
Data Output Rate from the FPGA

Since the front-end support is discussed, we briefly describe a possible scheme of supporting a front-end circuit via a 4-pair cable, although it is out of the scope of this document. In this scheme, a pair in cable is reserved to supply power to the front-end circuit board. In normal operation, a differential pair CC carries clock and commands using C5 scheme while the remaining two pairs are used to send trigger or DAQ data out. On power up, the FPGA is to be configured which is supported with a LVDS to TTL converter DS90C32B or similar device. Before the FPGA is configured, the converter is enabled and the FPGA pins are tri-stated that allows signals for configuration being sent via the differential pairs of CC and TRIG/DATA. After the FPGA is configured, the DS90C32B is disabled and all the differential pairs resume normal definitions.

2.5 Digital Phase Follower

Serial communication is a popular data transmission scheme since the communication channel is simply a twist-pair of a cable. There are serial transceivers available in several FPGA families with operating bit

rate higher than 1 Gb/s. However, the costs of FPGA with build-in transceivers are normally higher than the comparable devices without transceivers. There are applications of serial communications to be implemented in low-cost FPGA families where there are no dedicated transceivers, but relatively lower bit rates are sufficient. The Digital Phase Follower (DPF) is developed to fill this gap.

The transmitter of serial data is relatively simple which is a parallel to serial converter implemented either with a shift register or dual-port memory with single-bit output port. The receiver is more complicate since the cable delay causes the data to arrive at any possible phases. When temperature of the cable varies, the phase of the serial data may drift away from the original phase. If the clocks of the sender and the receiver are not derived from the same source, a continuous and indefinite phase drift is expected.

The DPF uses multiple samples of the data stream to detect and to keep track of the input data phase. In each bit time, the input data is sampled 4 times at 0, 90 180 and 270 degrees. The 4 clocks (or 0 and 90 degrees plus their inverted versions) can be generated with a PLL block now available in most low-cost FPGA families. The block diagram of the DPF is shown in Figure 2.9.

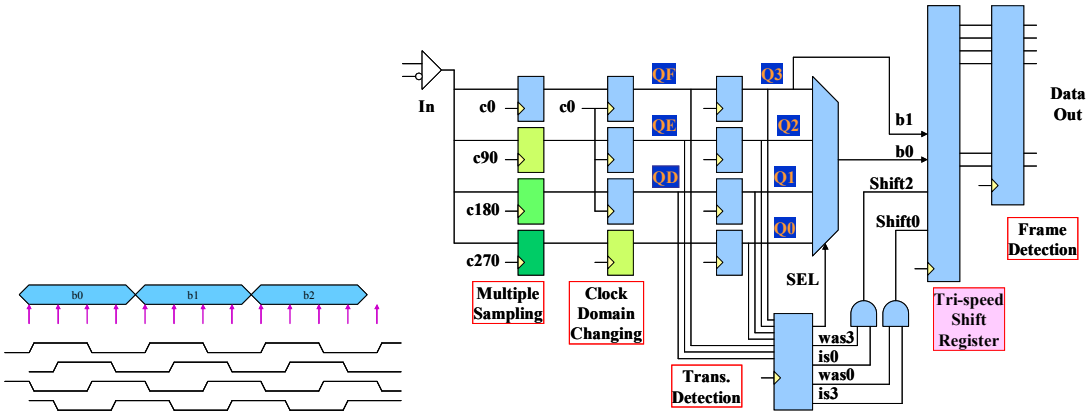

Figure 2.9
The Multi-sampling and Digital Phase Follower

The multi-sampling part is the same as in FPGA TDC discussed before. In fact, the operation of the DPF is based on the transition time of the input data stream. After multi-sampling, the sampled pattern is first converted to the 0 degree clock domain. Then 7 samples, QD to Q3 are sent to the transition detection logic to find the relative phase of the input data as shown in Figure 2.10. Note that the sample pattern jumps up 4 bits every clock cycles, i.e., QD jumps to Q1, QE to Q2 and QF to Q3 which is obvious from the pipeline structure shown in Figure 2.9.

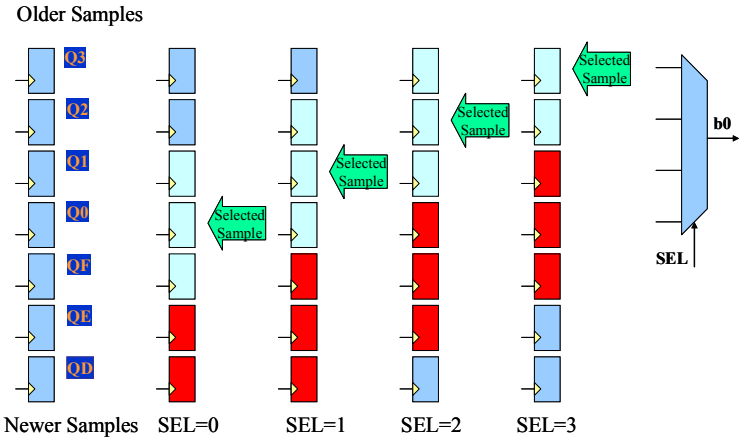

Figure 2.10
The Transition Detection in the Digital Phase Follower

The designers may choose to detect both 0-to-1 and 1-to-0 transitions, but detecting only one transition is recommended since the raising and falling time of the input circuit may be different. Once the first transition is seen, the location of it is registered and the data sample sufficiently far away from the transition points is selected as an input of the shift register in the later stage. For example, when a 0-to-1 transition is seen between QF and QE, i.e., (QF==0)&&(QE==1), the sample Q0 is selected, and so on.

When the phase of the input data drifts away from the original point, the transition at different location is detected. The sample point being selected follows the change of the transition location change accordingly.

As mentioned earlier, the input data phase may drift indefinitely if the clocks of the sender and receiver have very close but slightly different frequencies. Even in the systems with same clock source for the sender and receiver, the phase drift due to cable temperature variation may also be bigger than a bit time.

We can assume that the phase drift rate is not too high so that position of the current transition is either the same as what previously detected, or +1 or -1 from the previous position. Under this assumption, there are two possible cases for the bit phase drifting out of one bit time. The two cases: "was-0-is-3" and "was-3-is-0" are shown in Figure 2.11.

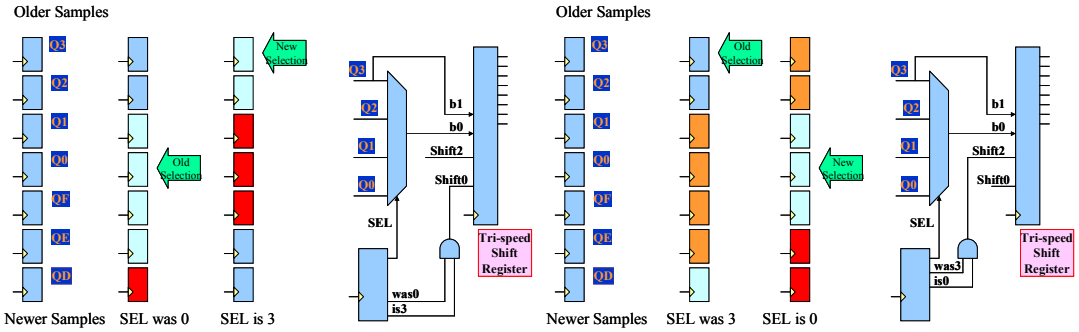

Figure 2.11
The Digital Phase Following Processes

The first case is finding the transition between Q2 and Q1 requesting that the sample Q3 being selected in current clock cycle while the previously registered selection was Q0. This was-0-is-3 case indicates that the input data clock is slower than the local clock. The selection point is actually drifted from Q0 to QF. However, to prevent the sampling point from drifting down indefinitely, it is wrapped over to Q3 instead of QF. Since the current sample at Q3 has been shifted into the shift-register in previous clock cycle (which was Q0), the shift register stops shifting for one clock cycle to compensate for the slower input data clock.

The second case is finding the transition between QF and QE requesting that the sample Q0 being selected in current clock cycle while the previously registered selection was Q3. This was-3-is-0 case indicates that the input data clock is faster than the local clock. The selection point is actually drifted from Q3 to Q4 (which is not implemented). However, to prevent the sampling point from drifting up indefinitely, it is wrapped over to Q0. In this situation, two sample points, i.e., Q3 and Q0 must be pushed into the shift register causing it shift by 2 bits in the current clock cycle to compensate for the faster input data.

The shift register normally shifts by 1 and it shifts by 0 or by 2 in the "was-0-is-3" or "was-3-is-0" cases, respectively, which is why it is called "tri-speed shift register". Typical de-serialization circuits, either in FPGA or single IC chip, recover the receiving clock using either PLL or clock swapping schemes. The digital phase follower is a pure digital circuit and the clock recovery is avoided.

3 A possible specification of the MIPP TDC card

3.1 TDC in FPGA

Many time measurement functions in high-energy/nuclear physics experiments can be implemented in FPGA directly. There are two types of practical TDC structures one may choose from as shown in Figure 2.1.

Figure 3.1
TDC structures in FPGA

The first structure shown in left uses chain structure like carry chain found in FPGA devices. The position of the input signal being registered represents the relative time difference between the input signal and the reference clock. The structure is commonly used in TDC ASIC chips except in ASIC chips the delay chain is adjusted by a control voltage that is derived by a feedback loop, so that the delay of each tap is a known constant. In FPGA, the delay of the delay chain is not controlled and it changes as the temperature and power supply voltage vary.